

DIAGRAMAS DE FLUJO: DFD

DFD es un programa de libre disposición para ayuda al diseño e implementación de algoritmos expresados en diagramas de flujo (DF). Además incorpora opciones para el depurado de los algoritmos, lo que facilita enormemente la localización de los errores de ejecución y lógicos más habituales.

Su utilización es muy sencilla, al tratarse de una herramienta gráfica, y además incluye un menú de ayuda muy completo, por lo que en estas notas nos vamos a centrar en el uso básico de las herramientas de diseño y depuración. El resto de opciones (detalles de sintaxis más avanzados, operadores y funciones disponibles), puede consultarse directamente en la ayuda del programa.

1. INICIO DE DFD

La ejecución de DFD presenta la pantalla de inicio siguiente



donde nos fijaremos en la barra de herramientas.



Aunque puede accederse a todas las opciones que comentaremos a continuación a través del menú, y con atajos de teclado, en estas notas las describiremos a través de los botones correspondientes.

- El bloque de **botones de objetos** nos permite seleccionar los distintos elementos (objetos) que vamos a introducir en el DF: sentencias de asignación, selección, iteración, ...
- El bloque de **ejecución** permite poner en funcionamiento el algoritmo
- El bloque de **depuración** se utiliza, en caso de funcionamiento incorrecto, para detectar errores en la construcción del algoritmo y corregirlos.
- Los botones de **subprogramas** permiten introducir funciones definidas por el programador
- Los restantes botones tienen una funcionalidad similar a la de las restantes aplicaciones Windows: abrir fichero, guardar fichero, cortar, pegar, ... Puede verse su tarea asociada acercando el cursor del ratón (sin hacer clic) al botón correspondiente.

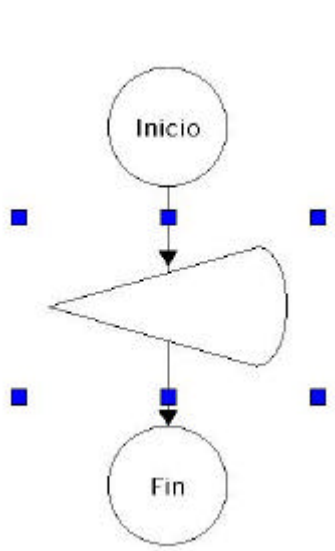
1.1 Un primer ejemplo de diseño con DF

Construiremos un primer ejemplo sencillo de algoritmo para ilustrar las capacidades más básicas de DFD. Dicho algoritmo consistirá en pedir un número al usuario y presentarlo por pantalla.

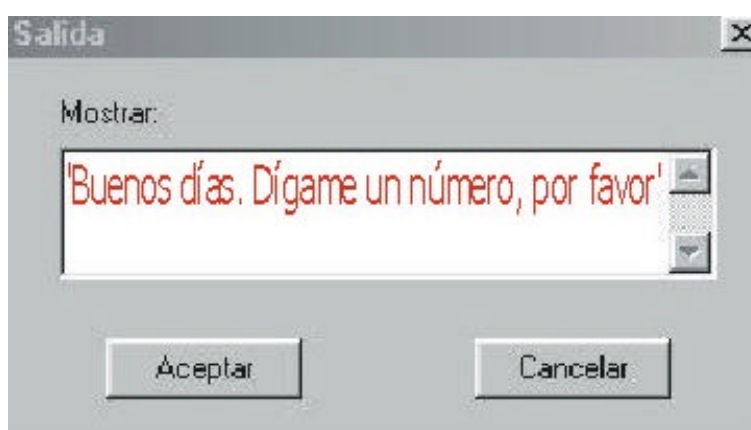
La operación básica será la de inserción de objetos. En primer lugar, insertaremos una sentencia de salida que le pida al usuario el número que posteriormente se va a imprimir. Para ello pulsamos el botón correspondiente al objeto que se desea insertar



y llevamos el ratón al punto donde vamos a insertarlo. La inserción se realiza pulsando el botón izquierdo, con lo que tendremos una situación como la siguiente:



Los puntos azules indican qué objeto se acaba de insertar. Para introducir en la sentencia de salida el mensaje que queremos imprimir será necesario EDITAR dicho objeto, haciendo doble clic sobre el mismo. De este modo se abre una ventana donde podemos dicho mensaje (por ejemplo 'Buenos días. Dígame un número, por favor').

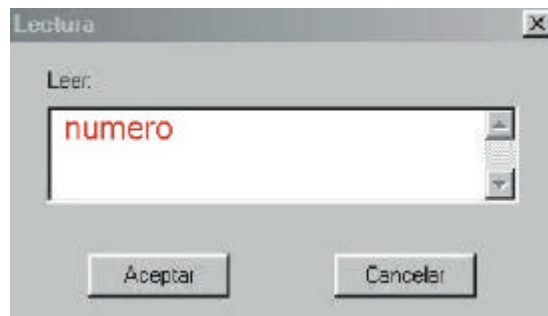


Como el mensaje es una cadena de caracteres, no debemos olvidarnos de las comillas simples al inicio y final de la misma.

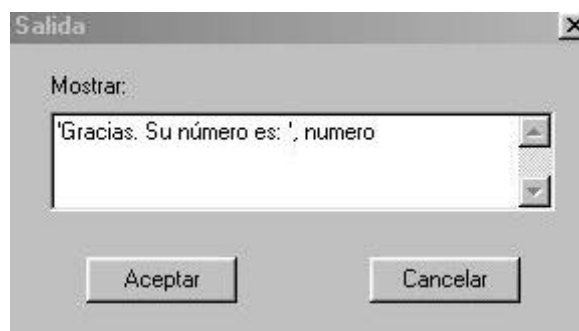
Seguidamente vamos a insertar una sentencia de ENTRADA, para almacenar en una variable el valor del número que nos proporcione el usuario. Para ello pulsaremos el botón correspondiente



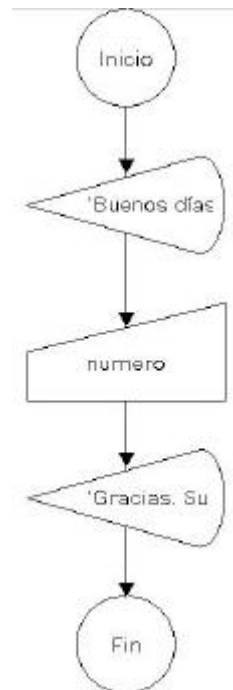
y lo insertaremos a continuación de la sentencia de salida anterior. Si editamos el objeto, haciendo doble clic sobre el mismo, aparecerá una pantalla cuyo cuadro de texto nos permitirá darle nombre a la variable donde vamos a guardar el valor (en este ejemplo la variable se va a llamar **numero**):



Para finalizar, mostraremos al usuario el número que ha introducido, para lo cual insertaremos una nueva sentencia de SALIDA, que editaremos para que muestre el siguiente mensaje:

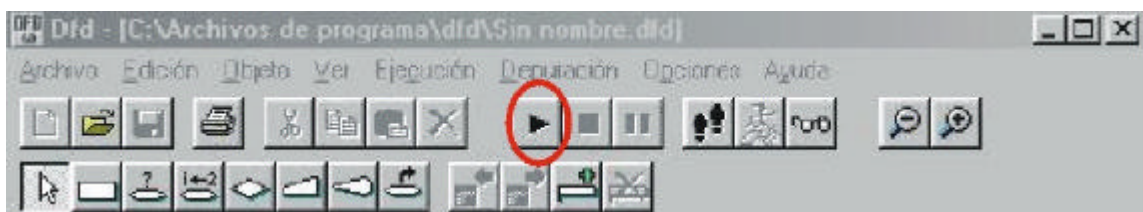


con lo que el algoritmo tendrá el siguiente aspecto en pantalla:



1.2 Un primer ejemplo de ejecución con DF

Tras haber diseñado el algoritmo podemos probar a ejecutarlo, al objeto de detectar posibles errores en él. Para ello utilizaremos los botones de ejecución, y en particular el botón EJECUTAR



que pondrá en marcha el algoritmo.

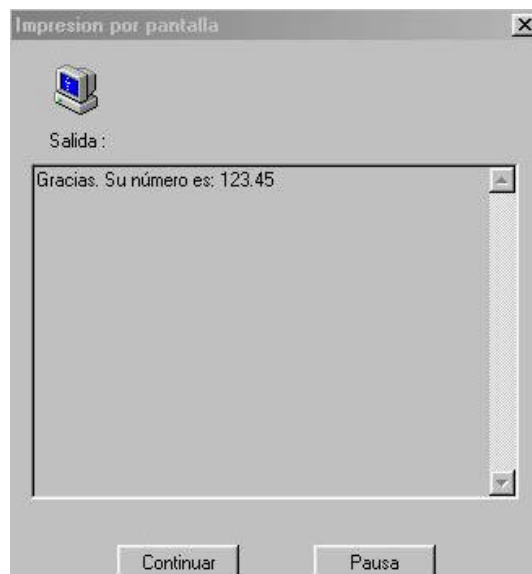
La primera sentencia en ejecutarse será la de SALIDA, que mostrará en pantalla el mensaje correspondiente:



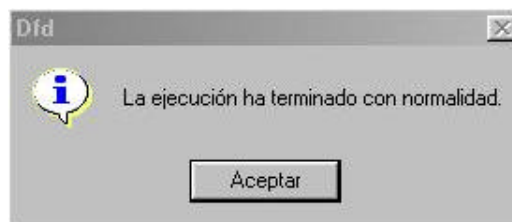
Seguidamente la de ENTRADA, que nos muestra un cuadro de texto donde introduciremos el valor que queramos darle a la variable (por ejemplo, 123.45):



y, finalmente, la última sentencia de SALIDA:



Cuando el algoritmo finaliza su ejecución sin error se muestra el siguiente mensaje:



Dado que el algoritmo es correcto, procederemos a guardarlo (por ejemplo, con el nombre **entradasalida**). La opción de guardar es similar a la de cualquier aplicación Windows, por lo que no merece mayor comentario. Únicamente recordar que en general, durante el proceso de elaboración de un algoritmo (que puede ser largo) debemos guardar frecuentemente en disco el diseño, al objeto de prevenir posibles fallos o errores que dejen inutilizado el ordenador y provoquen la pérdida del trabajo realizado.

Ejercicio 1: al objeto de ver ejemplos de errores, modificaréis el algoritmo anterior en el siguiente sentido:

1. errores de sintaxis: Eliminar una de las comillas en alguna de las sentencias de salida y ejecutar el algoritmo.
2. errores de ejecución: Eliminar la sentencia de entrada (para ello seleccionaréis dicha sentencia haciendo clic sobre el objeto y pulsáis el botón ELIMINAR o la tecla SUPRIMIR). Ejecutar el algoritmo.

Ejercicio 2: diseñar un nuevo algoritmo que pida al usuario dos números a y b y le diga cuál es su suma. Guardar.

2. OBJETOS DEFINIDOS EN DFD

DFD permite incluir los objetos básicos de programación estructurada: asignación, selección, lazos y subprogramas. Cualquier objeto que se inserte en el algoritmo puede ser editado haciendo doble clic, lo que permite definir los elementos que lo componen. Esto quiere decir que la EDICIÓN permitirá, por ejemplo, en el caso de:

- sentencias de salida: indicar la expresión que se va a presentar en pantalla

- sentencias de entrada: indicar los nombres de las variables donde se guardará la información
- sentencias de asignación: indicar las expresiones y los nombres de las variables donde se guardará el resultado
- estructuras de selección: indicar la condición
- ...

Otra acción interesante sobre los objetos es la **SELECCIÓN** de los mismos (clic sobre el objeto), que permite realizar acciones como eliminarlos y cortarlos o copiarlos para posteriormente pegarlos en otro punto del algoritmo.

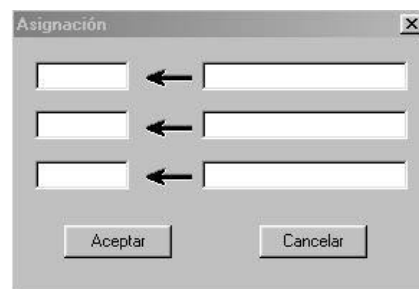
Veamos a continuación los aspectos más destacados a este respecto. Para mayor detalle, remitimos al menú de ayuda de DFD (tecla F1).

2.1 Sentencia de asignación

Se accede a ella con el botón



y su edición permite introducir hasta TRES asignaciones en la misma sentencia:



Para formar expresiones válidas tendremos en cuenta que DFD admite los siguientes elementos, todos ellos bien documentados en las opciones "Conceptos básicos (Tipos y conceptos de datos)" y "Referencia de operadores y funciones" del menú de ayuda:

- Constantes y variables
 - de tipo numérico
 - de tipo carácter (entre comillas simples)

- de tipo lógico (valores .V. y .F.)
- Operadores aritméticos habituales (+, -, *, /, ^), junto con otros como el operador módulo (MOD)
- Funciones matemáticas: logaritmos y exponenciales, trigonométricas, redondeo y truncamiento número (ROUND, TRUNC), ...
- Funciones de manejo de cadenas de caracteres: longitud de una cadena (LEN) y extracción de subcadenas (SUBSTRING)

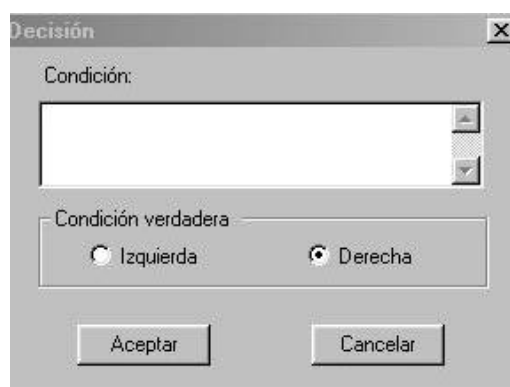
Ejercicio:

Diseñar y ejecutar un algoritmo que pida dos números **a** y **b** al usuario y calcule su suma, resta y producto. Guardarlo con el nombre **asignacion**.

2.2 Estructura de selección



Al editar el elemento una vez insertado se puede introducir la condición que se va a evaluar, a través de la ventana siguiente:



que también permite indicar qué rama (izquierda o derecha) va a corresponder al caso **CIERTO** de la condición. Al pulsar **ACEPTAR** en esta ventana, automáticamente se incluye el punto de confluencia de ambas ramas (punto de cierre de la estructura), que será el lugar por donde progrese el flujo del algoritmo una vez ejecutada la rama correspondiente.

Las condiciones en DFD son expresiones lógicas (que o bien son ciertas o bien falsas), que admiten los operadores habituales:

- Operadores de comparación: $>$, $<$, $>=$, $<=$, $=$, \neq
- Operadores lógicos: AND, OR, NOT

En cada una de las ramas se podrán insertar los objetos que se necesiten, igual que en cualquier otra parte del programa. En particular, se pueden insertar nuevas estructuras de selección para dar lugar a la estructura de selección múltiple. En todo momento DFD redibujará la estructura para mantener la legibilidad de la misma.

Ejercicios:

Diseñar y ejecutar un algoritmo que indique si un número **a** pedido por teclado es positivo o negativo. Guardarlo con el nombre **seleccion1**.

Modificar el algoritmo anterior para que considere también el caso en que **a** sea igual a cero. Guardarlo con el nombre **seleccion2**.

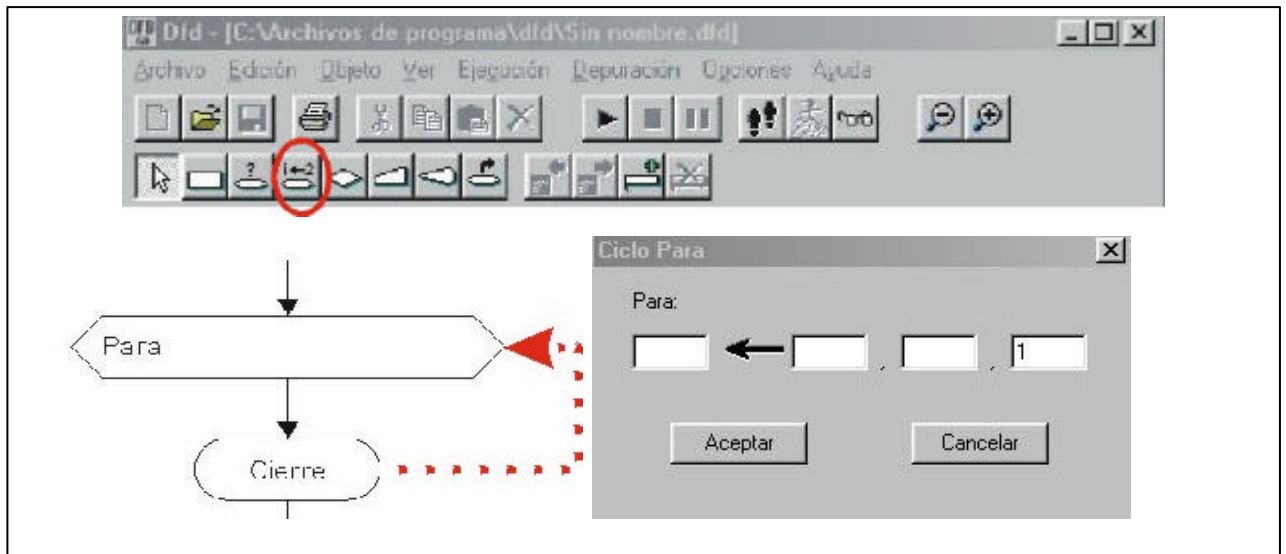
Modificar el algoritmo **asignacion** para que incluya la división, y que no produzca error de ejecución cuando **b** sea igual a cero.

2.3 Lazos

DFD permite dos tipos de lazos: el MIENTRAS y el DESDE (que en DFD se llama ciclo "para"), por lo que el lazo REPETIR-HASTA QUE debe ser diseñado a partir de los dos anteriores.

2.3.1 Lazo desde

La siguiente figura muestra el botón correspondiente al lazo DESDE, junto con su símbolo en DFD y la ventana de edición correspondiente. Cabe señalar que la representación DFD no utiliza flechas hacia atrás para indicar el final de la estructura, sino un indicador especial etiquetado como CIERRE.



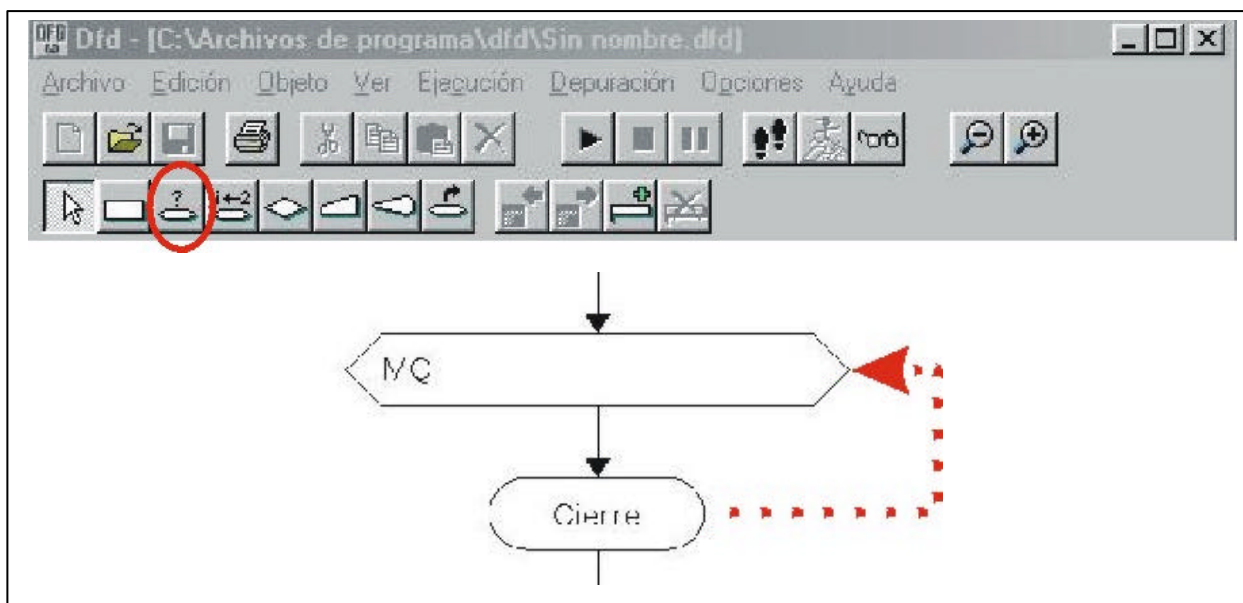
La ventana de edición permite indicar en su parte izquierda el nombre de la variable del lazo, y a la derecha los valores (enteros o reales) de inicio, final e incremento deseados.

Ejercicio

Diseñar y ejecutar un algoritmo que calcule el factorial de un número **n** pedido al usuario por teclado. Guardarlo con el nombre **factorial1**. Probarlo con valores $n=-1, 0, 1, 2$ y 100 .

2.3.2 Lazo mientras

En la figura se muestran el botón correspondiente al lazo mientras y su representación en DFD. La ventana de edición es idéntica a la de la estructura de selección, por lo que ya no la mencionamos.



El símbolo DFD tampoco utiliza la representación habitual de la flecha hacia atrás, como es habitual en la representación en DF, sino el símbolo de CIERRE.

Ejercicio

Modificar el algoritmo **factorial1**. para utilizar la estructura mientras. Guardarlo con el nombre **factorial2**.

3. AGRUPACIONES ESTÁTICAS DE DATOS: VECTORES Y MATRICES

DFD admite agrupaciones de datos, a las cuales denomina "arreglos" (fonéticamente similar a la palabra inglesa original, *array*).

Hay que señalar que la asignación de valores a una agrupación de datos debe hacerse siempre COMPONENTE A COMPONENTE, no pudiendo manejarse vectores o matrices completas. Esto debe tenerse en cuenta también para cualquier operación (entrada/salida, condiciones, lazos, ...).

La forma de referenciar un elemento de un vector o matriz es mediante su índice, que puede ser una constante, una variable o una expresión, pero siempre un valor ENTERO. El índice se expresa entre paréntesis, y habrá tantos índices como dimensiones tenga la agrupación de datos (1 para vectores, 2 en matrices, ...).

Así, las expresiones siguientes tienen el significado que se indica:

- $V(2)$ 2ª componente de un vector de nombre V

- $M(i, j)$ Componente ij de una matriz de nombre M

Es importante señalar que cuando se usen variables como índices para referenciar un elemento de una agrupación de datos, éstas deben tener un valor conocido en el momento de realizar la referencia. En caso contrario, se producirá un error.

Ejercicios:

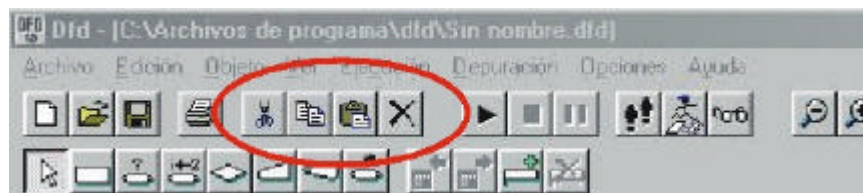
Pedir al usuario una lista de valores numéricos y calcular su suma.

Pedir al usuario dos matrices **A** y **B** de 2x2 elementos y calcular su resta.

En el ejercicio anterior, añadir una sentencia de salida que imprima los elementos $B(0,0)$, $B(3,3)$, $B(1,5)$. ¿Qué tipo de error se produce?

4. MANEJO DE OBJETOS

Los objetos DFD pueden eliminarse, copiarse o moverse de sitio siguiendo la estrategia habitual de Windows, con la salvedad de que sólo puede trabajarse con UN objeto en cada operación. Para ello se dispone de la barra de botones correspondiente

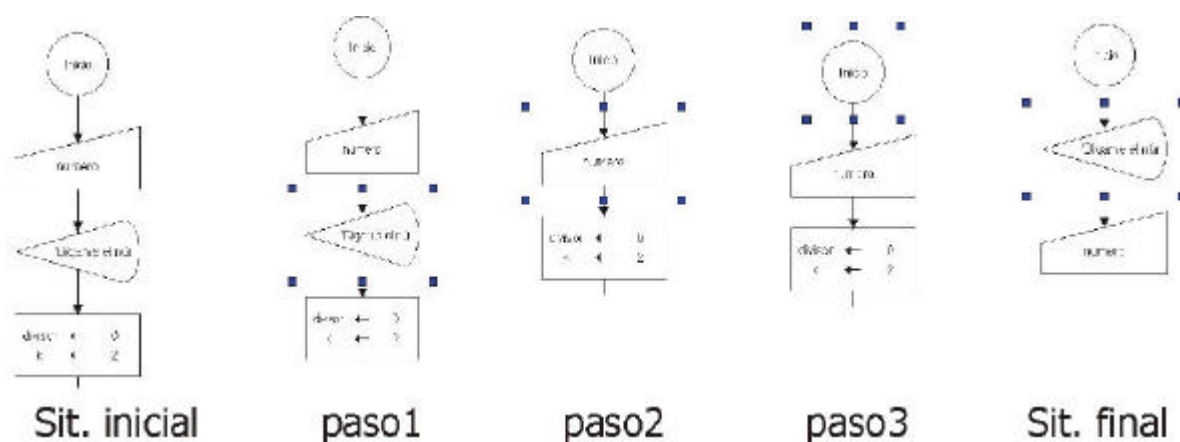


que actuará siempre sobre el objeto SELECCIONADO (clic sobre el mismo).

La única novedad destacable en las operaciones de copiar, eliminar y cortar es que DFD no dispone del botón DESHACER, por lo que deberán hacerse con cuidado, ya que un objeto eliminado o cortado no podrá recuperarse de ninguna forma que no sea definiéndolo manualmente de nuevo.

La operación PEGAR permite situar un objeto desde el portapapeles de Windows en cualquier punto del algoritmo. En DFD indicaremos el punto donde deseamos pegar un objeto del portapapeles seleccionando el objeto a continuación del cual deseamos colocarlo. Es decir, debemos tener en cuenta que el objeto va a quedar pegado JUSTO DESPUÉS del "objeto destino". La figura siguiente ilustra el proceso,

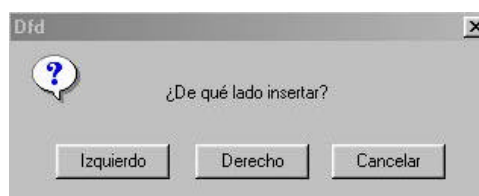
para un ejemplo en donde se pretende mover la sentencia de salida al inicio del algoritmo:



Para ello se opera del siguiente modo:

- Paso 1: Seleccionamos la sentencia que vamos a mover
- Paso 2: Pulsamos el botón de cortar
- Paso 3: Seleccionamos la sentencia que está justo antes de donde vamos a insertar el objeto cortado (sentencia de inicio)
- Sit. final: Pulsamos el botón pegar.

En caso de que el punto de destino sea la condición de una sentencia de selección, se abrirá una ventana que nos pedirá la rama donde deseamos colocar el objeto:



Estas operaciones pueden realizarse igualmente, como en cualquier aplicación Windows, con los contenidos de cualquier cuadro de texto (por ejemplo, al definir asignaciones, condiciones, lazos, ...).

5. DEPURACIÓN DE ALGORITMOS

Las herramientas de depuración se utilizan para detectar la/s sentencia/s en donde se han producido errores en el diseño de un algoritmo. La tarea de depuración consiste básicamente en explorar el algoritmo, ejecutándolo paso a paso y comparando en todo momento los valores que van tomando las distintas variables

con los valores esperados. Es necesario, por tanto, haber analizado con anterioridad algunos casos de prueba que permitan anticipar en todo momento los valores que deben tomar las variables y saber si la progresión del algoritmo es correcta o no.

Normalmente la depuración se realizará ante la presencia de errores de ejecución o lógicos, ya que los de sintaxis suelen ir acompañados de algún tipo de mensaje que facilita su localización.

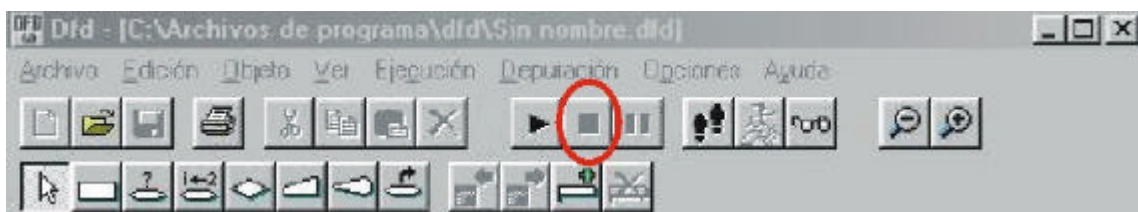
Los botones relacionados con la depuración son los siguientes (acercando el ratón a cada uno de ellos puedes ver la etiqueta con el nombre que le asocia DFD):



5.1 PASO SIMPLE

Permite ir ejecutando el algoritmo sentencia a sentencia. Combinado con la ventana de evaluación de variables permite ir viendo los valores que toman estas. A cada pulsación del botón, el flujo del algoritmo avanza una sentencia. El símbolo de la sentencia que se va a ejecutar en cada momento es destacado en color azul.

En cualquier momento puede pararse la ejecución paso a paso pulsando el botón DETENER:



Ejercicio

Ejecutar paso a paso el algoritmo de resta de dos matrices.

5.2 EJECUTAR HASTA

Esta opción es útil cuando se sabe con certeza que una parte del algoritmo está correctamente diseñada y por tanto, la ejecución paso a paso de dicha parte no es necesaria (además de poder ser bastante tediosa).

El botón EJECUTAR HASTA permite establecer un PUNTO DE RUPTURA en las sentencias del algoritmo, de modo que el programa se ejecutará con normalidad hasta dicho punto, y a partir de ahí puede realizarse alguna de las siguientes posibilidades:

- evaluar variables
- retomarse la ejecución normal
- seguir paso a paso
- establecer un punto de ruptura en una sentencia posterior del algoritmo y continuar hasta él

El punto de ruptura debe establecerse con anterioridad, seleccionando (clic) la sentencia donde se quiere fijar éste, y seguidamente pulsando el botón EJECUTAR HASTA. En el momento que la ejecución alcance el punto de ruptura, el algoritmo se detendrá, y la sentencia correspondiente queda marcada en color azul.

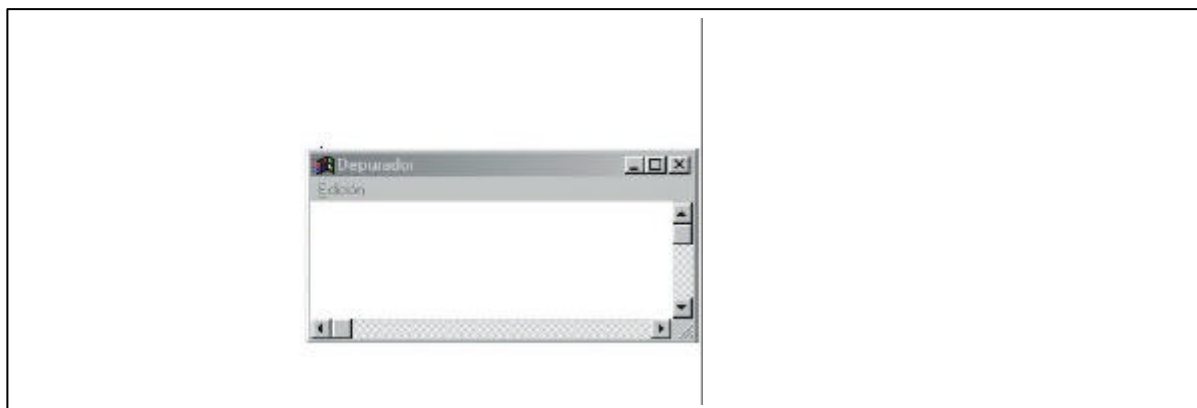
Ejercicio

Establecer un punto de ruptura en la primera sentencia ejecutable del algoritmo de resta de dos matrices, y continuar paso a paso a partir de ahí.

5.3 DEPURADOR

Abre una ventana donde se pueden escribir los nombres de las variables (o expresiones en general) cuyo valor se desea explorar durante la depuración del algoritmo. Se utilizará en combinación con cualquiera de los dos métodos de depuración anteriores, para valorar si las variables toman valores correctos o no.

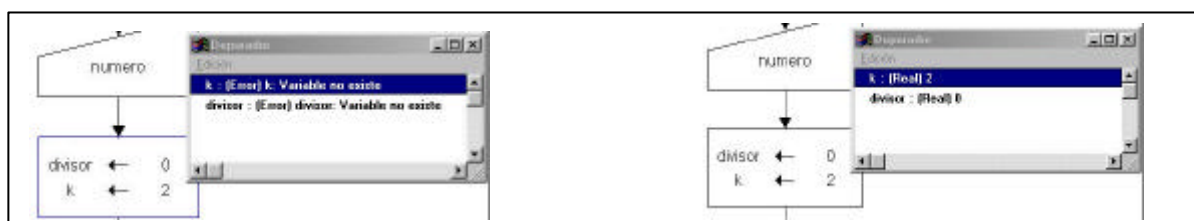
Al pulsar el botón correspondiente se abre una ventana en la que se van a visualizar las variables o expresiones que interese evaluar.



Dichas expresiones se introducen en el cuadro de texto que aparece pulsando la tecla INSERTAR, cuando la ventana del depurador está activada (si no lo estuviera, basta con hacer clic en cualquier punto de ella para activarla).

Si se desea eliminar alguna de las expresiones, basta con marcarla con el ratón (clic) y pulsar la tecla SUPRIMIR. Esto puede resultar útil, ya que una vez insertada una variable en la ventana del depurador

En una sesión de depuración, normalmente se tendrá visible la ventana del depurador, con el objeto de ir comparando en todo momento los valores que toman las variables. Cuando una variable no ha sido inicializada, se mostrará el mensaje "Variable no existe", que cambiará al valor correspondiente una vez ejecutada la sentencia que le asigna un valor (sentencia de asignación o sentencia de entrada). La siguiente figura ilustra esta situación:




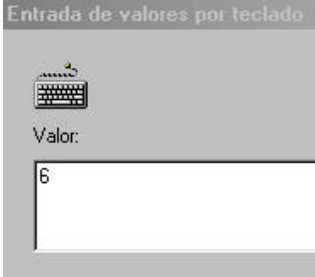
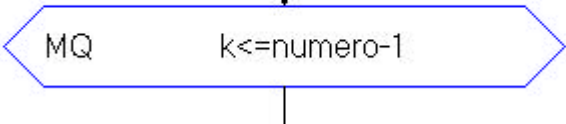



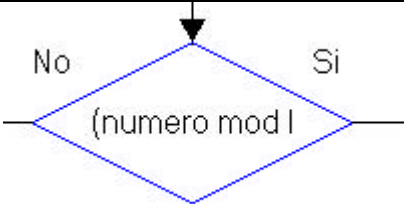
Puede verse en la parte izquierda cómo la sentencia que inicializa las variables no se ha ejecutado aún (señalada en azul), y por tanto las variables no toman valores conocidos. Una vez ejecutada, las variables toman los valores esperados.


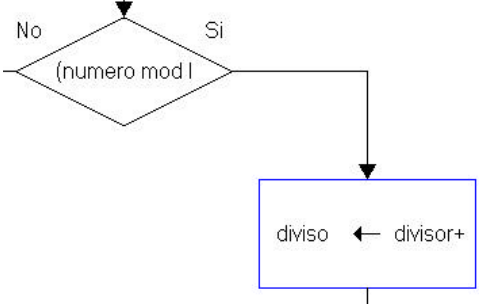

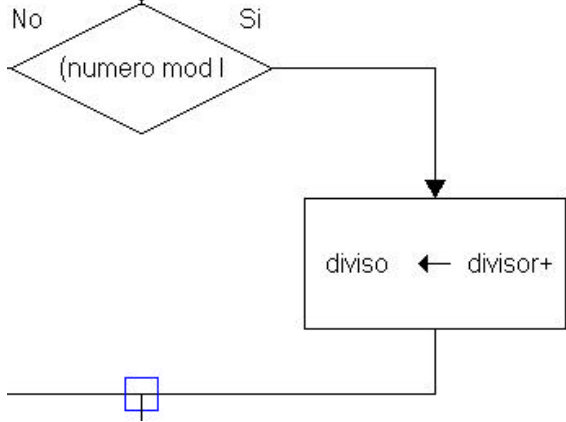

5.4 Depuración de errores en DFD. Sesión de ejemplo.

En primer lugar, copiaremos del Web de prácticas el fichero **numeroprimo.dfd**, disponible en la sección Metodología de la Programación. El algoritmo presenta un error lógico, como podéis comprobar ejecutándolo para algunos casos (2, 4, 15, ...).

Para detectar dicho error realizaremos una ejecución hasta el lazo mientras, ya que no parece probable que el error esté en las primeras sentencias del programa.

Por tanto, fijamos el punto de ruptura en dicha sentencia, seleccionándola y pulsando el botón EJECUTAR HASTA. A partir de aquí, el proceso evoluciona de la siguiente manera:

Acción	Consecuencia	Resultado
	Se nos pide el número. Introducimos el valor 6:	
	La ejecución alcanza el punto de ruptura	
 + INS	Abrimos una ventana del depurador para evaluar las variables definidas hasta el momento. En este momento, todas ellas toman valores correctos	
	A partir de aquí seguimos paso a paso. La siguiente instrucción es el mientras, que se debe verificar.	

	<p>Se cumple la condición, ya que 2 es divisor de 6. Por tanto, ya sabemos que el número NO es primo, y divisor debe incrementarse y pasar a valer 1.</p>	
	<p>Se alcanza el final de la estructura de selección, y la variable divisor sigue valiendo cero. ¡Aquí hay un error!</p>	
	<p>Detenemos la ejecución paso a paso para editar la asignación errónea y corregirla.</p>	

Una vez corregido el error, probamos nuevamente el algoritmo con los casos de prueba anteriores, verificando que funciona correctamente.

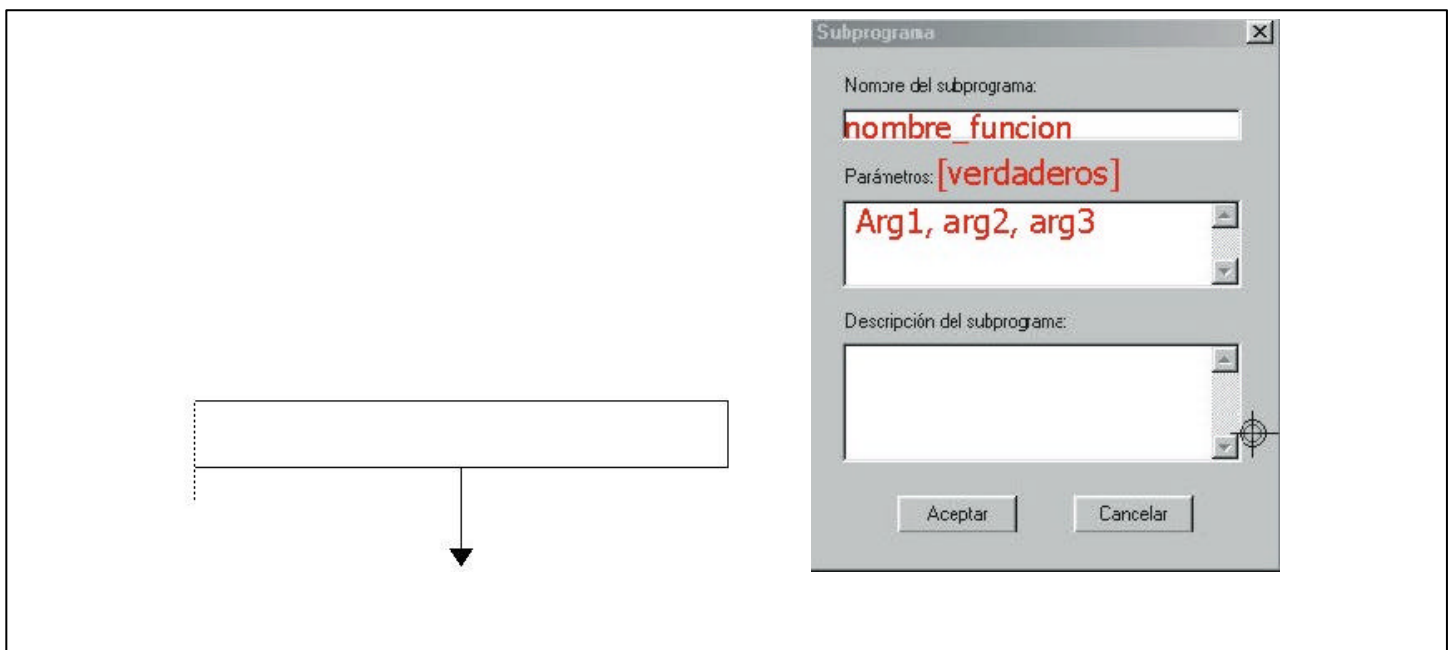
6. SUBPROGRAMAS

El tipo de subprograma que admite DFD es la función. Debe notarse que en DFD los argumentos se pasan por REFERENCIA cuando son nombres de variables (tanto variables escalares como vectores o matrices), y se pasan por VALOR cuando son expresiones. Dentro de la opción OBJETOS del menú de ayuda, las opciones LLAMADA y SUBPROGRAMAS profundizan en estos aspectos. Igualmente, las funciones DFD no disponen de la sentencia DEVOLVER, por lo que los valores que deba retornar la función deberán almacenarse en los correspondientes argumentos de salida.

El código correspondiente a las funciones incluidas en un algoritmo debe añadirse al mismo utilizando el botón NUEVO SUBPROGRAMA



que abre una nueva pantalla similar a la de inicial de un algoritmo, en la cual cambia el símbolo INICIO por la cabecera de la función. Editando dicha cabecera se establecerá el nombre de la función, los parámetros (ficticios) correspondientes y, si se desea, una breve descripción de dicha función (documentación de la misma).

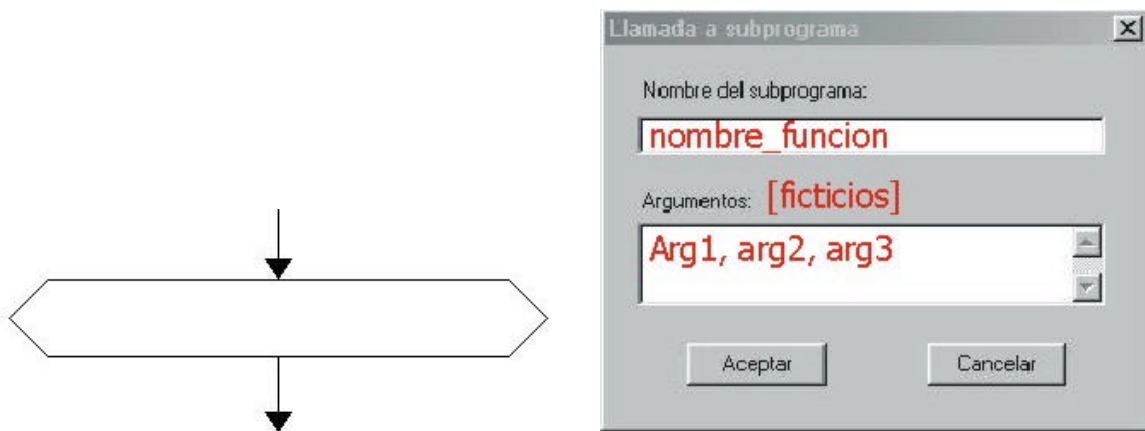


La inserción, borrado y edición de objetos en una función es idéntica a como se describió para el algoritmo principal.

La llamada a una función desde otro módulo tiene también un símbolo especial en DFD



cuya edición permite introducir el nombre y los parámetros verdaderos de la función:



Los restantes botones del menú subprogramas permiten moverse entre las distintas ventanas de subprogramas o eliminar el subprograma actual:



Por último, señalar que las operaciones de COPIAR, CORTAR, PEGAR y ELIMINAR pueden realizarse entre subprogramas de un mismo algoritmo y entre algoritmos que se abran en una misma sesión de DFD.

Esta última posibilidad puede aprovecharse para reutilizar funciones previamente construidas sin necesidad de volver a construirlas manualmente en el nuevo algoritmo. Dado que desafortunadamente DFD no permite copiar y pegar un algoritmo o función completa, es preciso recurrir a un pequeño artificio para poder realizar esta tarea de forma fácil.

El artificio consiste en encerrar las funciones que diseñemos con DFD en una sentencia que englobe totalmente a la función (salvo la cabecera), y que no añada nada a la ejecución de la misma (por ejemplo, un lazo desde que se ejecute una sola vez). De esta manera sí es posible copiar dicho lazo desde (y por tanto, la función

completa), y pegarla en otro algoritmo donde vayamos a utilizar la función. Lo único que habrá que completar manualmente será la cabecera, que como ya hemos mencionado no puede copiarse.